



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/705,525	11/10/2003	Attila Barta	RSW920030176US1	5400
51016	7590	11/21/2006	EXAMINER	
IBM CORP. (RALEIGH SOFTWARE GROUP) c/o Rudolf O Siegesmund Gordon & Rees, LLP 2100 Ross Avenue Suite 2600 DALLAS, TX 75201			CHEN, QING	
		ART UNIT		PAPER NUMBER
		2191		

DATE MAILED: 11/21/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	10/705,525	BARTA ET AL.	
	Examiner	Art Unit	
	Qing Chen	2191	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1) Responsive to communication(s) filed on 10 October 2006.
 2a) This action is FINAL. 2b) This action is non-final.
 3) Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4) Claim(s) 1,3-14,16-21 and 23-40 is/are pending in the application.
 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
 5) Claim(s) _____ is/are allowed.
 6) Claim(s) 1,3-14,16-21 and 23-40 is/are rejected.
 7) Claim(s) _____ is/are objected to.
 8) Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9) The specification is objected to by the Examiner.
 10) The drawing(s) filed on 20 September 2006 is/are: a) accepted or b) objected to by the Examiner.
 Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
 Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
 11) The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12) Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
 a) All b) Some * c) None of:
 1. Certified copies of the priority documents have been received.
 2. Certified copies of the priority documents have been received in Application No. _____.
 3. Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) <input checked="" type="checkbox"/> Notice of References Cited (PTO-892)	4) <input type="checkbox"/> Interview Summary (PTO-413)
2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948)	Paper No(s)/Mail Date. _____
3) <input type="checkbox"/> Information Disclosure Statement(s) (PTO/SB/08)	5) <input type="checkbox"/> Notice of Informal Patent Application
Paper No(s)/Mail Date _____	6) <input type="checkbox"/> Other: _____

DETAILED ACTION

1. This Office action is in response to the amendment filed on October 10, 2006.
2. **Claims 1, 3-14, 16-21, and 23-40** are pending.
3. **Claims 1, 3-14, 16-21, 23, 25-28, 34, 39, and 40** have been amended.
4. **Claims 2, 15, and 22** have been cancelled.
5. The objections to the drawings are withdrawn in view of Applicant's amendments to the drawings.
6. The objections to the specification are withdrawn in view of Applicant's amendments to the specification.
7. The objections to Claims 20, 21, 39, and 40 are withdrawn in view of Applicant's amendments to the claims.
8. The 35 U.S.C. § 101 rejections of Claims 1-19, 21, 22, 28, 32, and 34-40 are withdrawn in view of Applicant's amendments to the claims. However, the 35 U.S.C. § 101 rejection of Claim 20 is maintained and further explained below.

Response to Amendment

Claim Objections

9. **Claims 1, 3-14, 16-19, 21, 23-34, and 40** are objected to because of the following informalities:

- **Claim 1** contains a typographical error: a semicolon (;) and the word "and" should be added after the first limitation.

- **Claims 1, 14, and 21** contain a typographical error: the word “and” should be deleted after the limitation “... fourth plurality of components suitable for parallel installation.”
- **Claims 1, 14, and 21** recite the limitation “... a seventh plurality of data ...” The limitation should presumably read -- a seventh plurality of metadata --.
- **Claims 3-13, 16-19, and 23-33** depend on Claims 1, 14, and 21, respectively, and, therefore, suffer the same deficiency as their respective parent claims.
- **Claim 4** contains a typographical error: the phrase “... providing a user with a plurality of options *if* for the eight plurality of potential conflicts” should presumably read “...providing a user with a plurality of options for the eight plurality of potential conflicts.”
- **Claim 13** contains a typographical error: the word “components” in the phrase “... the second components” in the last limitation should be changed to singular form.
- **Claim 14** contains the following typographical errors:
 - A semicolon (;) should be added after the first limitation.
 - The period (.) after the word “system” in the second limitation should be deleted.
 - The comma (,) at the end of the second limitation should be changed to a semicolon (;).
- **Claim 14** should presumably recite the word “comprising” after the second limitation, since the “means-plus-function” limitations recited afterwards are elements of the program installed in memory.
- **Claims 16-19** depend on Claim 14 and, therefore, suffer the same deficiencies as Claim 14.

- **Claim 16** depends on Claim 15. However, Claim 15 has been cancelled. The Examiner subsequently interprets this claim as depending on Claim 14 for the purpose of further examination.
- **Claim 23** depends on Claim 22. However, Claim 22 has been cancelled. The Examiner subsequently interprets this claim as depending on Claim 21 for the purpose of further examination.
- **Claim 34** contains a typographical error: the word “and” should be added after the second-to-last limitation.
- **Claim 40** recites the limitation “the component compatibility data structure.” The Examiner subsequently interprets this limitation as reading “the data structure” for the purpose of providing it with proper explicit antecedent basis.
Appropriate correction is required.

Claim Rejections - 35 USC § 112

10. The following is a quotation of the second paragraph of 35 U.S.C. 112:

The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

11. **Claims 1, 3-14, 16-19, 21, and 23-33** are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Claims 1, 8, 12, 21, and 28 recite the limitation “the system.” There is insufficient antecedent basis for this limitation in the claims. In the interest of compact prosecution, the Examiner subsequently interprets this limitation as reading “a system” for the purpose of further examination.

Claims 3-7 depend on Claim 1 and, therefore, suffer the same deficiency as Claim 1.

Claims 9-11 depend on Claim 8 and, therefore, suffer the same deficiency as Claim 8.

Claim 13 depends on Claim 12 and, therefore, suffers the same deficiency as Claim 12.

Claims 29-33 depend on Claim 28 and, therefore, suffer the same deficiency as Claim 28.

Claim 14 recites the limitation “a system and a program installed in a memory of the computer.” It is unclear to the Examiner how a system—supposedly comprising of hardware elements—could be installed on a memory of the computer. In the interest of compact prosecution, the Examiner subsequently interprets this limitation as reading “a program installed in a memory of the computer” for the purpose of further examination.

Claim 14 recites the limitation “the pre-deployment analysis.” There is insufficient antecedent basis for this limitation in the claim. In the interest of compact prosecution, the Examiner subsequently interprets this limitation as reading “a pre-deployment analysis” for the purpose of further examination.

Claims 16-19 depend on Claim 14 and, therefore, suffer the same deficiencies as Claim 14.

Claims 21 and 23 recite the limitation “the data structure.” There is insufficient antecedent basis for this limitation in the claims. In the interest of compact prosecution, the Examiner subsequently interprets this limitation as reading “a data structure” for the purpose of further examination.

Claims 24-33 depend on Claim 21 and, therefore, suffer the same deficiencies as Claim 21.

Claim Rejections - 35 USC § 101

12. 35 U.S.C. 101 reads as follows:

Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

13. **Claim 20** is rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

It is noted that **Claim 20**, merely claimed as a data structure, contains functional descriptive material *per se*, and therefore, is held non-statutory. Data structures not claimed as embodied in computer-readable media are descriptive material *per se*, and are not statutory

because they are not capable of causing functional change in the computer. See, e.g., *Warmerdam*, 33 F.3d at 1361, 31 USPQ2d at 1760 (claim to a data structure *per se*, held non-statutory). Such claimed data structures do not define any structural and functional interrelationships between the data structure and other claimed aspects of the invention, which permit the data structure's functionality to be realized. In contrast, a claimed computer-readable medium encoded with a data structure defines structural and functional interrelationships between the data structure and the computer software and hardware components, which permit the data structure's functionality to be realized, and is thus statutory.

Claim Rejections - 35 USC § 102

14. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

15. **Claims 20, 34, 35, and 37-40** are rejected under 35 U.S.C. 102(b) as being anticipated by Curtis (US 6,442,754).

As per **Claim 20**, Curtis discloses:

- for each of the second plurality of software components, a component deployment dependency data, an indication of necessary components for an operation of each of the second plurality of software components, and an indication of incompatibility with one or more

components of the first plurality of components (see *Figure 5; Column 13: 7-27 and 33-37*, “*a data structure ... is maintained in the registry object or registry database 220, indicating installed programs and dependent components on which installed programs depend.*” and “*Each installed file set component has a Dependency subdirectory which includes information on each dependent component on which the file set and program depend in order to operate. The dependency subdirectory would list the program name, version, fileset name, and fileset version for each program on which the fileset including the dependency subdirectory depends.*” and “*...the dependency directory may indicate dependent file sets or registry objects that are the subject matter of the processed dependency object. If there are no dependent components, then the dependency directory will contain no values.*”); and

- wherein an alert is automatically generated if an attempt is made to install a component having an indication of incompatibility (see *Column 10: 36-40*, “*The description ('desc') field 434 provides a written description of the dependency. The description in the field 434 is displayed to the user if the dependency is not satisfied. The description information may list the name of the dependent component(s) not installed.*”).

As per **Claim 34**, Curtis discloses:

- loading an installation package, the installation package including a data structure (see *Figure 2: 101; Column 5: 56-58*, “*A script, referred to herein as 'installerjava', 101 FIG. 2, is used to run the install engine. The script implements the base installer class in Java.*”); *Column 7: 40-45*, “*... a platform independent registry database class 220 is created which ties the*

platform specific code 201 with registry objects 332. The registry database 220 implements the registry function for those platforms which do not have a registry. ");

- searching a target to which components are to be installed to identify a plurality of previously installed components (*see Column 11: 11-20, "... a call to the check_dependency function ... This function determines whether the file, program or registry object indicated in the dependency object 400 is installed on the computer. ";*)
 - for a first component, accessing, in the data structure, a component deployment dependency data, an indication of necessary components for an operation of the first component, and an indication of incompatibility with a previously installed component (*see Figure 5; Column 13: 7-27 and 33-37, "... a data structure ... is maintained in the registry object or registry database 220, indicating installed programs and dependent components on which installed programs depend." and "Each installed file set component has a Dependency subdirectory which includes information on each dependent component on which the file set and program depend in order to operate. The dependency subdirectory would list the program name, version, files set name, and files set version for each program on which the files set including the dependency subdirectory depends." and "...the dependency directory may indicate dependent file sets or registry objects that are the subject matter of the processed dependency object. If there are no dependent components, then the dependency directory will contain no values. ";*)

and

- analyzing a plurality of data from the data structure to determine a plurality of conflicts between the first component to be installed and the plurality of components previously installed on the system (*see Column 12: 27-32, "... generate a list from the information*

maintained in the description ('desc') field 434 of all returned dependency objects to display to the user, indicating dependent components that must be installed before the depending program may be installed.").

As per **Claim 35**, the rejection of **Claim 34** is incorporated; and Curtis further discloses:

- notifying a user of the conflict (*see Column 10: 36-40, "The description ('desc') field 434 provides a written description of the dependency. The description in the field 434 is displayed to the user if the dependency is not satisfied. The description information may list the name of the dependent component(s) not installed."*).

As per **Claim 37**, the rejection of **Claim 34** is incorporated; and Curtis further discloses:

- ignoring a detected conflict and continuing the installation (*see Column 12: 51-53, "When the user selects to install the file, the install program 17 will execute the install script in either the Install 416 or SInstall field 418."*).

As per **Claim 38**, the rejection of **Claim 37** is incorporated; and Curtis further discloses:

- entering a note in a log of the conflict (*see Figure 2: 140; Column 7: 4-5, "During install, the log 140 and 'uninstall.Java1' 150 information are built."; Column 8: 24-29, "... providing various logs, e.g. a log for keeping track of what is being installed, and a log that reports the progress of install. Logs are used for both the install and uninstall process. Furthermore, these logs are human readable which allows them to be checked, e.g., after a silent install, to ensure that a file has installed successfully."*).

As per **Claim 39**, the rejection of **Claim 34** is incorporated; and Curtis further discloses:

- initiating the removal of an installed component (*see Figure 6: 560; Column 13: 50-51, "... the program processes a request to uninstall a program. ");*
- accessing the data structure (*see Column 13: 51-52, "Control transfers to block 562 where the program processes all the dependency directories. ");* and
- identifying a conflict if the installed component is removed (*see Column 13: 63-67, "... displays to the user on the display means 14 information indicating the depending programs that should be uninstalled before continuing with the uninstallation of the program, which is a dependent program..").*

As per **Claim 40**, the rejection of **Claim 34** is incorporated; and Curtis further discloses:

- initiating an installation of a second component (*see Figure 2: 340; Column 11: 23-24, "... a file set 340 is installed. ");*
- searching a target to which the second component is to be installed to verify installed components (*see Column 11: 11-20, "... a call to the check_dependency function ... This function determines whether the file, program or registry object indicated in the dependency object 400 is installed on the computer. ");*
- accessing the data structure (*see Column 11: 50-52, "... the check_dependency function may utilize a registry database 220 to maintain information typically maintained in a registry file. ");* and

Art Unit: 2191

- determining if all of the components required by the second component are installed

(see Column 11: 57-61, "... determine whether the program and version indicated in the Program Name 408 and Program Version 410 fields is installed in the system 10. ").

Claim Rejections - 35 USC § 103

16. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

17. **Claims 1, 3, 4, 6-14, 16-19, 21, 23, 24, 26-33** are rejected under 35 U.S.C. 103(a) as being unpatentable over Curtis (US 6,442,754) in view of Foster (US 6,675,382).

As per **Claim 1**, Curtis discloses:

- using a data structure in a storage that provides, for each of the plurality of software components, a component deployment dependency data, an indication of necessary components for an operation of each of the plurality of software components being installed, and an indication of incompatibility with a previous installed component (see Figure 5; Column 13: 7-27 and 33-37, "... a data structure ... is maintained in the registry object or registry database 220, indicating installed programs and dependent components on which installed programs depend." and "Each installed file set component has a Dependency subdirectory which includes information on each dependent component on which the file set and program depend in order to

operate. The dependency subdirectory would list the program name, version, fileset name, and fileset version for each program on which the fileset including the dependency subdirectory depends." and "...the dependency directory may indicate dependent file sets or registry objects that are the subject matter of the processed dependency object. If there are no dependent components, then the dependency directory will contain no values. ");

- using a computer connected to the storage and a program installed in a memory of the computer (see *Figure 1: 10; Column 5: 29-31, "The programs in memory 12 includes an operating system (OS) 16 program and application programs, such as an install program 17 or an installer tool kit."*), performing the steps of:

- determining a first plurality of components previously installed on a system (see *Column 11: 11-20, "... a call to the check_dependency function ... This function determines whether the file, program or registry object indicated in the dependency object 400 is installed on the computer.*");

- determining a second plurality of components to be installed on a system (see *Figure 2: 340; Column 11: 23-24, "... a file set 340 is installed.*");

- accessing a third plurality of components deployment dependency data (see *Column 13: 18-21, "Each installed file set component has a Dependency subdirectory which includes information on each dependent component on which the file set and program depend in order to operate.*");

- accessing a sixth plurality of metadata from the data structure regarding the second plurality of components to be installed and accessing a seventh plurality of metadata regarding the first plurality of components previously installed (see *Column 13:*

13-15 and 21-24, "A root directory includes a sub-directory for each installed program, indicating the program name and version." and "The dependency subdirectory would list the program name, version, fileset name, and fileset version for each program on which the fileset including the dependency subdirectory depends."); and

- analyzing the sixth plurality of metadata to determine an eight plurality of potential conflicts between the second plurality of components to be installed and the first plurality of components previously installed on a system (*see Column 12: 27-32, "...generate a list from the information maintained in the description ('desc') field 434 of all returned dependency objects to display to the user, indicating dependent components that must be installed before the depending program may be installed."*).

However, Curtis does not disclose:

- determining a fourth plurality of components suitable for parallel installation;
- determining an order in which the fourth plurality of components can be grouped for a fifth plurality of parallel installations; and
- wherein the pre-deployment analysis allows the second plurality of components to be installed in parallel and in sequence of groups, whereby an installation time is reduced.

Foster discloses:

- determining a fourth plurality of components suitable for parallel installation (*see Column 10: 6-8, "... other packages may be concurrently installed that require the presence of package 200 on the system."*);
- determining an order in which the fourth plurality of components can be grouped for a fifth plurality of parallel installations (*see Column 10: 8-10, "... the system checks the*

Art Unit: 2191

dependencies between package 200 and the packages that are being simultaneously installed."); and

- wherein the pre-deployment analysis allows the second plurality of components to be installed in parallel and in sequence of groups, whereby an installation time is reduced (*see Column 10: 8-10, "... the packages that are being simultaneously installed.").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Foster into the teaching of Curtis to include determining a fourth plurality of components suitable for parallel installation; determining an order in which the fourth plurality of components can be grouped for a fifth plurality of parallel installations; and wherein the pre-deployment analysis allows the second plurality of components to be installed in parallel and in sequence of groups, whereby an installation time is reduced. The modification would be obvious because one of ordinary skill in the art would be motivated to provide an efficient and simple solution for packaging, distributing and installing software (*see Foster – Column 1: 43-44).*

As per **Claim 3**, the rejection of **Claim 1** is incorporated; and Curtis further discloses:

- updating the data structure with an identity of a ninth plurality of recently installed components (*see Column 13: 28-30, "The information in this directory is created whenever a component is installed. For instance, whenever a program is installed, a subdirectory is created under the root directory. ".*)

As per **Claim 4**, the rejection of **Claim 1** is incorporated; and Curtis further discloses:

Art Unit: 2191

- providing a user with a plurality of options for the eight plurality of potential conflicts (*see Column 12: 35-45, "If so, control transfers to block 534 where the program displays on the display means 14 the name of the dependent component that was not located on the system and a radio button to allow the user to selectively cause the execution of the install script in the Install 416 or SInstall 418 fields. If there is not install script, then control transfers to block 536 where the program displays information maintained in the install information field 426 to inform the user on where to obtain the dependent component that is needed before the program may be installed.").*

As per **Claim 6**, the rejection of **Claim 4** is incorporated; and Curtis further discloses:

- wherein a second option includes continuing an installation (*see Column 12: 51-53, "When the user selects to install the file, the install program 17 will execute the install script in either the Install 416 or SInstall field 418. "*).

As per **Claim 7**, the rejection of **Claim 6** is incorporated; and Curtis further discloses:

- upon the exercise of the second option, recording an entry in a log indicative of a conflict and of a continuation of installation (*see Figure 2: 140; Column 7: 4-5, "During install, the log 140 and 'uninstall.Java1' 150 information are built."; Column 8: 24-29, "... providing various logs, e.g. a log for keeping track of what is being installed, and a log that reports the progress of install. Logs are used for both the install and uninstall process. Furthermore, these logs are human readable which allows them to be checked, e.g., after a silent install, to ensure that a file has installed successfully. "*).

As per **Claim 8**, the rejection of **Claim 1** is incorporated; and Curtis further discloses:

- initiating a removal of component from a system (*see Figure 6: 560; Column 13: 50-51, "... the program processes a request to uninstall a program."*); and
- identifying a tenth plurality of remaining components which depend on the component to be removed (*see Column 13: 59-62, "The uninstall program may navigate the directory structure from the dependency directory shown in FIG. 5 to determine dependant programs that depend on the program subject to the uninstallation."*).

As per **Claim 9**, the rejection of **Claim 8** is incorporated; and Curtis further discloses:

- providing a user with a plurality of options if the tenth plurality of dependent remaining components is identified (*see Column 12: 35-45, "If so, control transfers to block 534 where the program displays on the display means 14 the name of the dependent component that was not located on the system and a radio button to allow the user to selectively cause the execution of the install script in the Install 416 or SInstall 418 fields. If there is not install script, then control transfers to block 536 where the program displays information maintained in the install information field 426 to inform the user on where to obtain the dependent component that is needed before the program may be installed."*).

As per **Claim 10**, the rejection of **Claim 9** is incorporated; and Curtis further discloses:

- wherein a first option includes aborting a removal (*see Figure 6: 570; Column 13: 62-63, "Control then transfers to block 570 to exit uninstallation ... "*).

As per **Claim 11**, the rejection of **Claim 9** is incorporated; and Curtis further discloses:

- wherein a second option includes continuing a removal (*see Figure 6: 568; Column 13: 55-56, “Otherwise, control transfers to block 568 to proceed with the uninstallation.”*).

As per **Claim 12**, the rejection of **Claim 8** is incorporated; and Curtis further discloses:

- identifying a first component previously installed on a system which is dependent upon a removed component (*see Column 13: 4-6 and 64-67, “... before uninstalling a program, a determination may be made as to whether other installed components depend on the file being uninstalled.” and “... information indicating the depending programs that should be uninstalled before continuing with the uninstallation of the program, which is a dependent program.”*); and
- determining an identity of a second component upon which the first component depends (*see Column 13: 1-4, “During installation of a dependent program, dependency information from the Dependency Object 400 may be written to a dependency location indicating dependent components of the installed file.”*).

As per **Claim 13**, the rejection of **Claim 12** is incorporated; and Curtis further discloses:

- installing the second component upon which the first component depends (*see Column 13: 1-4, “During installation of a dependent program, dependency information from the Dependency Object 400 may be written to a dependency location indicating dependent components of the installed file.”*); and

- creating a dependency link between the first component and the second component

(see Column 13: 1-4, "Dependency Object 400 may be written to a dependency location indicating dependent components of the installed file.").

Claim 14 is a system claim corresponding to the method claim above (Claim 1) and, therefore, is rejected for the same reason set forth in the rejection of Claim 1.

As per **Claim 16**, the rejection of **Claim 14** is incorporated; and Curtis further discloses:

- a means for loading an installation package including the data structure (see Figure 2: 101; Column 5: 56-58, "A script, referred to herein as 'installerjava', 101 FIG. 2, is used to run the install engine. The script implements the base installer class in Java."; Column 7: 40-45, "... a platform independent registry database class 220 is created which ties the platform specific code 201 with registry objects 332. The registry database 220 implements the registry function for those platforms which do not have a registry.").

As per **Claim 17**, the rejection of **Claim 14** is incorporated; and Curtis further discloses:

- a ninth plurality of references among the components to be installed and located in the data structure (see Column 13: 18-24, "Each installed file set component has a Dependency subdirectory which includes information on each dependent component on which the file set and program depend in order to operate. The dependency subdirectory would list the program name, version, filesset name, and filesset version for each program on which the filesset including the dependency subdirectory depends.").

As per **Claim 18**, the rejection of **Claim 17** is incorporated; and Curtis further discloses:

- a means for accessing the data structure (*see Column 13: 28-29, “The information in this directory is created whenever a component is installed.”*).

As per **Claim 19**, the rejection of **Claim 14** is incorporated; and Curtis further discloses:

- a means for installing the second plurality of components across a plurality of enterprise resources (*see Column 4: 39-44, “... the dependency object may be used across operating systems to check dependencies for all operating systems on which the install program operates. In this way, the dependency object is part of the cross-platform capabilities that allow the installer program to install products on different operating system platforms.”*).

As per **Claim 21**, Curtis discloses:

- determining a first plurality of components previously installed on a system (*see Column 11: 11-20, “... a call to the check_dependency function ... This function determines whether the file, program or registry object indicated in the dependency object 400 is installed on the computer.”*);
- determining a second plurality of components to be installed on a system (*see Figure 2: 340; Column 11: 23-24, “... a file set 340 is installed.”*);
- accessing a third plurality of components deployment dependency data (*see Column 13: 18-21, “Each installed file set component has a Dependency subdirectory which includes*

information on each dependent component on which the file set and program depend in order to operate.”);

- accessing a sixth plurality of metadata from a data structure regarding the second plurality of components to be installed and accessing a seventh plurality of metadata regarding the first plurality of components previously installed (*see Column 13: 13-15 and 21-24, “A root directory includes a sub-directory for each installed program, indicating the program name and version.” and “The dependency subdirectory would list the program name, version, fileset name, and fileset version for each program on which the fileset including the dependency subdirectory depends.”*); and
- analyzing the sixth plurality of metadata to determine an eight plurality of potential conflicts between the second plurality of components to be installed and the first plurality of components previously installed on a system (*see Column 12: 27-32, “...generate a list from the information maintained in the description ('desc') field 434 of all returned dependency objects to display to the user, indicating dependent components that must be installed before the depending program may be installed.”*).

However, Curtis does not disclose:

- determining a fourth plurality of components suitable for parallel installation;
- determining an order in which the fourth plurality of components can be grouped for a fifth plurality of parallel installations; and
- wherein the pre-deployment analysis allows the second plurality of components to be installed in parallel and in sequence of groups, whereby an installation time is reduced.

Foster discloses:

- determining a fourth plurality of components suitable for parallel installation (see

Column 10: 6-8, "... other packages may be concurrently installed that require the presence of package 200 on the system.");

- determining an order in which the fourth plurality of components can be grouped for a fifth plurality of parallel installations (see *Column 10: 8-10, "... the system checks the dependencies between package 200 and the packages that are being simultaneously installed."*);

and

- wherein the pre-deployment analysis allows the second plurality of components to be installed in parallel and in sequence of groups, whereby an installation time is reduced (see *Column 10: 8-10, "... the packages that are being simultaneously installed."*).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Foster into the teaching of Curtis to include determining a fourth plurality of components suitable for parallel installation; determining an order in which the fourth plurality of components can be grouped for a fifth plurality of parallel installations; and wherein the pre-deployment analysis allows the second plurality of components to be installed in parallel and in sequence of groups, whereby an installation time is reduced. The modification would be obvious because one of ordinary skill in the art would be motivated to provide an efficient and simple solution for packaging, distributing and installing software (see Foster – *Column 1: 43-44*).

As per **Claim 23**, the rejection of **Claim 21** is incorporated; and Curtis further discloses:

Art Unit: 2191

- updating a data structure with an identity of a ninth plurality of recently installed components (*see Column 13: 28-30, "The information in this directory is created whenever a component is installed. For instance, whenever a program is installed, a subdirectory is created under the root directory. "*).

As per **Claim 24**, the rejection of **Claim 21** is incorporated; and Curtis further discloses:

- providing a user with a plurality of options if a conflict is identified (*see Column 12: 35-45, "If so, control transfers to block 534 where the program displays on the display means 14 the name of the dependent component that was not located on the system and a radio button to allow the user to selectively cause the execution of the install script in the Install 416 or SInstall 418 fields. If there is not install script, then control transfers to block 536 where the program displays information maintained in the install information field 426 to inform the user on where to obtain the dependent component that is needed before the program may be installed. "*).

As per **Claim 26**, the rejection of **Claim 24** is incorporated; and Curtis further discloses:

- wherein a second option includes continuing an installation (*see Column 12: 51-53, "When the user selects to install the file, the install program 17 will execute the install script in either the Install 416 or SInstall field 418. "*).

As per **Claim 27**, the rejection of **Claim 26** is incorporated; and Curtis further discloses:

- upon the exercise of the second option, recording an entry in a log indicative of the conflict and of a continuation of the installation (*see Figure 2: 140; Column 7: 4-5, "During*

install, the log 140 and ‘uninstall.Java1’ 150 information are built.”; Column 8: 24-29, “... providing various logs, e.g. a log for keeping track of what is being installed, and a log that reports the progress of install. Logs are used for both the install and uninstall process. Furthermore, these logs are human readable which allows them to be checked, e.g., after a silent install, to ensure that a file has installed successfully.”).

As per **Claim 28**, the rejection of **Claim 21** is incorporated; and Curtis further discloses:

- initiating a removal of a component from a system (*see Figure 6: 560; Column 13: 50-51, “... the program processes a request to uninstall a program.”*); and
- identifying a plurality of remaining components which depend on the component to be removed (*see Column 13: 59-62, “The uninstall program may navigate the directory structure from the dependency directory shown in FIG. 5 to determine dependant programs that depend on the program subject to the uninstallation.”*).

As per **Claim 29**, the rejection of **Claim 28** is incorporated; and Curtis further discloses:

- providing a user with a plurality of options if a dependent remaining component is identified (*see Column 12: 35-45, “If so, control transfers to block 534 where the program displays on the display means 14 the name of the dependent component that was not located on the system and a radio button to allow the user to selectively cause the execution of the install script in the Install 416 or SInstall 418 fields. If there is not install script, then control transfers to block 536 where the program displays information maintained in the install information field*

426 to inform the user on where to obtain the dependent component that is needed before the program may be installed.").

As per **Claim 30**, the rejection of **Claim 29** is incorporated; and Curtis further discloses:

- wherein a first option includes aborting the removal (*see Figure 6: 570; Column 13: 62-63, "Control then transfers to block 570 to exit uninstallation ... "*).

As per **Claim 31**, the rejection of **Claim 29** is incorporated; and Curtis further discloses:

- wherein a second option includes continuing the removal (*see Figure 6: 568; Column 13: 55-56, "Otherwise, control transfers to block 568 to proceed with the uninstallation."*).

As per **Claim 32**, the rejection of **Claim 28** is incorporated; and Curtis further discloses:

- identifying a first component previously installed on the system which is dependent upon a removed component (*see Column 13: 4-6 and 64-67, "... before uninstalling a program, a determination may be made as to whether other installed components depend on the file being uninstalled." and "... information indicating the depending programs that should be uninstalled before continuing with the uninstallation of the program, which is a dependent program."*); and
- indicating the identity of a second component upon which the first component depends (*see Column 13: 1-4, "During installation of a dependent program, dependency information from the Dependency Object 400 may be written to a dependency location indicating dependent components of the installed file."*).

As per **Claim 33**, the rejection of **Claim 32** is incorporated; and Curtis further discloses:

- installing the second component upon which the first component depends (see

Column 13: 1-4, "During installation of a dependent program, dependency information from the Dependency Object 400 may be written to a dependency location indicating dependent components of the installed file."); and

- creating a dependency link between the first and second components (see *Column 13: 1-4, "Dependency Object 400 may be written to a dependency location indicating dependent components of the installed file."*).

18. **Claims 5 and 25** are rejected under 35 U.S.C. 103(a) as being unpatentable over Curtis (US 6,442,754) in view of Foster (US 6,675,382) as applied to Claims 4 and 24 above, and further in view of Bourke-Dunphy et al. (US 6,918,112).

As per **Claim 5**, the rejection of **Claim 4** is incorporated; however, Curtis and Foster do not disclose:

- wherein a first option includes aborting an installation.

Bourke-Dunphy et al. disclose wherein a first option includes aborting an installation (see *Figure 5: 236; Column 8: 35-38, "... the user may select a CANCEL action button 236 to return to the component selection user interface ... where the user may manually modify the component selections."*).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Bourke-Dunphy et al. into the teaching of

Art Unit: 2191

Curtis to include wherein a first option includes aborting an installation. The modification would be obvious because one of ordinary skill in the art would be motivated to allow the user to exit the current installation, correct the error identified, and reinitiate the installation procedure (*see Bourke-Dunphy et al. – Column 1: 27-34*).

As per **Claim 25**, the rejection of **Claim 24** is incorporated; however, Curtis and Foster do not disclose:

- wherein a first option includes aborting an installation.

Bourke-Dunphy et al. disclose wherein a first option includes aborting an installation (*see Figure 5: 236; Column 8: 35-38, "... the user may select a CANCEL action button 236 to return to the component selection user interface ... where the user may manually modify the component selections."*).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Bourke-Dunphy et al. into the teaching of Curtis to include wherein a first option includes aborting an installation. The modification would be obvious because one of ordinary skill in the art would be motivated to allow the user to exit the current installation, correct the error identified, and reinitiate the installation procedure (*see Bourke-Dunphy et al. – Column 1: 27-34*).

19. **Claim 36** is rejected under 35 U.S.C. 103(a) as being unpatentable over Curtis (US 6,442,754) in view of Bourke-Dunphy et al. (US 6,918,112).

As per **Claim 36**, the rejection of **Claim 34** is incorporated; however, Curtis does not disclose:

- aborting the installation if a conflict is detected.

Bourke-Dunphy et al. disclose aborting the installation if a conflict is detected (*see*

Figure 5: 236; Column 8: 35-38, "... the user may select a CANCEL action button 236 to return to the component selection user interface ... where the user may manually modify the component selections.").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Bourke-Dunphy et al. into the teaching of Curtis to include aborting the installation if a conflict is detected. The modification would be obvious because one of ordinary skill in the art would be motivated to allow the user to exit the current installation, correct the error identified, and reinitiate the installation procedure (*see Bourke-Dunphy et al. – Column 1: 27-34*).

Response to Arguments

20. Applicant's arguments filed on October 10, 2006 have been fully considered but they are not persuasive.

In the remarks, Applicant argues that:

a) Curtis does not disclose determination of conflicts. Curtis states: "Block 530 represents the beginning of a loop to generate a list from the information maintained in the description

(“desc”) field 434 of all returned dependency objects to display to the user, indicating dependent components that must be installed before the depending program may be installed.” (12:27-32).

Examiner's response:

a) Curtis clearly discloses determination of conflicts. It is to be noted that the list from the information maintained in the description (“desc”) field of all returned dependency objects, indicating dependent components that must be installed before the depending program may be installed, is based on determination of conflicts (*see Column 12: 27-32*).

In the remarks, Applicant argues that:

b) Moreover, Curtis addresses the problem of installing a single program using dependency objects, but does not address a system and method for analyzing multiple programs for pre-deployment analysis in order to streamline the deployment by parallel and group processing. Thus, Curtis does not identify applications previously installed, but rather only checks for dependency objects related to the single installation of an application (10:60-65) and for the operating system on which the application is being installed (11:20-22).

Examiner's response:

b) Applicant's arguments fail to comply with 37 CFR § 1.111(b) because they amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the references.

Conclusion

21. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the Examiner should be directed to Qing Chen whose telephone number is 571-270-1071. The Examiner can normally be reached on Monday through Thursday from 7:30 AM to 4:00 PM. The Examiner can also be reached on alternate Fridays.

If attempts to reach the Examiner by telephone are unsuccessful, the Examiner's supervisor, Wei Zhen, can be reached on 571-272-3708. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the TC 2100 Group receptionist whose telephone number is 571-272-2100.

Art Unit: 2191

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).



WEI ZHEN
SUPERVISORY PATENT EXAMINER

QC / QC

November 13, 2006